01-27-00 A

UTILITY PATENT APPLICATION TRANSMITTAL

Submit an original and a duplicate for fee processing

(Only for new nonprovisional applications under 37 CFR §1.53(b))

jc760 U.S. PTO 01/24/00

jc584 U.S. PTO 09/490981 01/24/00

ADDRESS TO:

Assistant Commissioner for Patents
Box Patent Application
Washington, D.C. 20231

Attorney Docket No. 202269

First Named Inventor Raghuraman

Express Mail No. EL305741254US

APPLICATION ELEMENTS	ACCOMPANYING APPLICATION PARTS
1. ☒ Utility Transmittal Form	8. ☒ Assignment Papers (cover sheet and document(s))
2. ☒ Specification (including claims and abstract) [Total Pages 17]	9. ☐ Power of Attorney
3. ☒ Drawings [Total Sheets 5]	10. ☐ English Translation Document (if applicable)
4. ☒ Combined Declaration and Power of Attorney [Total Pages 3]	11. ☐ Information Disclosure Statement (IDS)
a. ☒ Newly executed	☐ Form PTO-1449
b. ☐ Copy from prior application **[Note Box 5 below]**	☐ Copies of References
i. ☐ Deletion of Inventor(s) Signed statement attached deleting inventor(s) named in the prior application	12. ☐ Preliminary Amendment
5. ☐ Incorporation by Reference: The entire disclosure of the prior application, from which a copy of the oath or declaration is supplied under Box 4b is considered as being part of the disclosure of the accompanying application and is hereby incorporated by reference therein.	13. ☒ Return Receipt Postcard (Should be specifically itemized)
6. ☐ Microfiche Computer Program	14. ☐ Small Entity Statement(s)
7. ☐ Nucleotide and/or Amino Acid Sequence Submission	☐ Enclosed
a. ☐ Computer Readable Copy	☐ Statement filed in prior application; status still proper and desired
b. ☐ Paper Copy	15. ☐ Certified Copy of Priority Document(s)
c. ☐ Statement verifying above copies	16. ☒ Other: Declaration of Venkat Ramanathan; Declaration of Melur Raghuraman; Check in the amount of $1060.00

17. **If a CONTINUING APPLICATION**, check appropriate box and supply the requisite information in (a) and (b) below:

(a) ☐ Continuation ☐ Divisional ☐ Continuation-in-part of prior application Serial No. . Prior application information: Examiner ; Group Art Unit:

(b) Preliminary Amendment: Relate Back - 35 USC §120. The Commissioner is requested to amend the specification by inserting the following sentence before the first line: "This is a ☐ continuation ☐ divisional of copending application(s)
☐ Serial No. , filed on .
☐ International Application, filed on , and which designates the U.S."

APPLICATION FEES

BASIC FEE				$690.00
CLAIMS	NUMBER FILED	NUMBER EXTRA	RATE	
Total Claims	21 -20=	1	x $18.00	$18.00
Independent Claims	7 - 3=	4	x $78.00	$312.00
☐ Multiple Dependent Claims(s) if applicable			+$260.00	$
			Total of above calculations =	$1020.00
			Reduction by 50% for filing by small entity =	$( )
☒ Assignment fee if applicable			+ $40.00	$40.00
			TOTAL =	$1060.00

UTILITY PATENT APPLICATION TRANSMITTAL	Attorney Docket No. 202269

19. ☐ Please charge my Deposit Account No. 12-1216 in the amount of $.

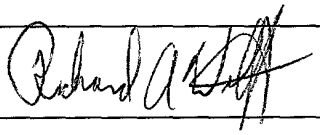
20. ☒ A check in the amount of $1060.00 is enclosed.

21. The Commissioner is hereby authorized to credit overpayments or charge any additional fees of the following types to Deposit Account No. 12-1216:
    a. ☒ Fees required under 37 CFR §1.16.
    b. ☒ Fees required under 37 CFR §1.17.

22. ☒ The Commissioner is hereby generally authorized under 37 CFR §1.136(a)(3) to treat any future reply in this or any related application filed pursuant to 37 CFR §1.53 requiring an extension of time as incorporating a request therefor, and the Commissioner is hereby specifically authorized to charge Deposit Account No. 12-1216 for any fee that may be due in connection with such a request for an extension of time.

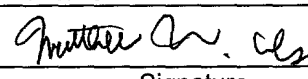
23. CORRESPONDENCE ADDRESS

Richard A. Wulff, Registration No. 42,238
Leydig, Voit & Mayer, Ltd.
Two Prudential Plaza, Suite 4900
180 North Stetson
Chicago, Illinois 60601-6780
Telephone: (312) 616-5600
Facsimile: (312) 616-5700

Name	Richard A. Wulff
Signature	
Date	January 24, 2000

Certificate of Mailing Under 37 CFR §1.10

I hereby certify that this Utility Patent Application Transmittal and all accompanying documents are being deposited with the United States Postal Service "Express Mail Post Office To Addressee" Service under 37 CFR §1.10 on the date indicated below and is addressed to: Assistant Commissioner for Patents, Box Patent Application, Washington, D.C. 20231.

MATTHEW OLSON		January 24, 2000
Name of Person Signing	Signature	Date

METHOD OF TRACING DATA TRAFFIC ON A NETWORK

CROSS REFERENCE TO RELATED APPLICATIONS

This application claims the benefit of U.S. Provisional Application No. 60/141,581, filed on June 23, 1999.

TECHNICAL FIELD OF THE INVENTION

This invention relates generally to event tracing and more particularly to a method of tracing data traffic on a network.

BACKGROUND

In this client/server world of computing that we live in today, capacity planning activity is just not limited to the server but to the entire system including the clients, servers and network devices. In the past, network capacity planners have treated the systems without serious concern and system capacity planners have in turn ignored the network devices in their analysis. One major reason for this is the lack of appropriate performance data to tie the two worlds and the fear of overhead of instrumentation.

Accuracy of model input metrics is often cited as the key indicator of the validity of a capacity analysis. In the last couple of years, there have been several papers mentioning the lack of performance data in MICROSOFT WINDOWS NT brand operating system mainly for Capacity Planning. These concerns were addressed by introducing an Event Tracing Facility in the WINDOWS NT 5.0 brand operating system and published the event tracing API. While it addressed system data requirements adequately, data related to network was lacking.

With internet access becoming common place today, there is no drop in the appetite for network bandwidth for web based applications. In fact, high speed networking is a very important focus of WINDOWS NT 5.0 brand operating system which already achieves 1 to 2 Gbps throughput. With network speeds getting this fast, any instrumentation must be highly optimized to take minimal overhead.

Most capacity planning efforts for networks have treated the system as a source generator and focused on frame counts and frame bytes by listening to the wire. Some have employed smart ways of identifying the application responsible for network traffic by scanning the packet headers. These methods are expensive and arbitrary.

SUMMARY OF THE INVENTION

To solve the aforesaid problems, a method of tracing data traffic on a network is provided herein. According to the method, trace instrumentation of the TCP/IP stack provides key information for capacity planners for correctly charging network traffic to the individual services and applications. The TCP/IP stack is instrumented at the transport layer, so that Input/Output Request packets (IRP) representing sends and receives can be detected as they pass through the stack. When such packets are detected an appropriate send or receive event is recorded in a trace log.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 shows the network layers of the WINDOWS NT brand operating system;

Fig. 2 shows the server and workstation services;

Fig. 3 shows the TCP/IP stack;

Fig. 4 is a timeline of a TCP send; and

Fig. 5 is a timeline of a TCP receive.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Event Tracing in WINDOWS NT 5.0 brand operating system

The next version of MICROSOFT WINDOWS NT brand operating system operating system will have a uniform framework for event tracing, specifically for capacity planning. The event tracing mechanism implements a circular buffer pool maintained by the operating system and an API set for Trace Providers, Consumers and Management Control. The trace logger can accept data from kernel mode device drivers and user mode applications.

In addition to providing a facility to log trace data for applications, the WINDOWS NT brand operating system kernel has been instrumented to provide key

capacity planning metrics that were not available through the commonly used performance tool 'Perfmon'. The following system events are instrumented:

1. Process Creation/Deletion event. The ProcessId, parent process Id, Security Id and the Image File name are recorded.

2. Thread Creation/Deletion event. The Thread Id and its process Id are also recorded.

3. Hard Page fault event. The disk signature and the size of the first disk read resulting from the page fault are also recorded.

4. Disk Read/Write event. The disk signature and the size of the operation are recorded.

Multiple logger streams may be active at one time, typically one for the kernel logger and one for each of the trace-enabled applications running on a server. The Consumer APIset makes it easy to process the trace from multiple logger streams in the proper order and returned to the caller one event at a time.

Networking Architecture in WINDOWS NT brand operating system

Networking capabilities are built into WINDOWS NT brand operating system and it is organized as layers as shown in Fig. 1.

WINDOWS NT brand operating system networking components include:

- Transport protocols (DLC 10, NetBEUI 12, NWLink, and TCP/IP 14) define the rules governing communications between two computers.
- Inter-process communication (IPC) components, such as named pipes and mail slots, allow applications to communicate with each other over a network.
- File and Print sharing components allow resources to be made available on a network.

The Multiple uniform naming convention (UNC) Provider (MUP) and Multi-Provider Router (MPR) make it possible to write applications that use a single API to communicate using any network vendor's redirector.

There are two boundary layers in the architecture, namely the Network Driver Interface Specification (NDIS) 16 and Transport Driver Interface (TDI) 18. The NDIS layer provides the interface and a wrapper to the Network Interface Card (NIC) device drivers 20. The TDI boundary layer 18 provides a common interface specification to communicate with various transport drivers. While several protocols are supported in the transport layer, TCP/IP 14 forms the main focal point for all networking activity.

The protocol suite benefits from years of research and is the most favored suite in the Internet. In the WINDOWS NT brand operating system, several services make use of TCP/IP stack, most notably File/Print services and Socket-based applications. Most services require reliable data transmission and use TCP/IP suite for end to end reliable delivery. The File/Print services and Sockets will be explained in more detail in the next section.

File and Print Services

The File and Print services are supported by two services (Redirector and Server) that are layered on top of Transport Driver Interface layer 18 as shown in Figure 2. They provide an encapsulation over the file system and network transparency for applications accessing remote files.

When a process on a WINDOWS NT brand operating system system tries to open a file that resides on a remote computer, the following steps occur:

- The process calls the I/O Manager to request that the file be opened.
- The I/O Manager recognizes that the request is for a file on a remote computer, so it passes it to the redirector file system driver 22 (RDR.Sys).
- The redirector 22 passes the request to lower-level network drivers that transmit it to the remote Server 26 for processing.
- The transport receives a send IO request packet (Irp) for the SMB header and command.
- This send is translated into frames and queued for dispatch on the wire through the Miniport driver.

On the remote WindowsNT server system, when the server service receives a request from a remote computer asking it to read a file that resides on the hard disk, the following steps occur:

- The low-level network drivers receive the request and pass it to the server driver 24 (SRV.SYS).
- The server 26 passes a file read request to the appropriate local file system driver.
- The local file system driver calls lower-level disk device drivers to access the file.
- The data is passed back to the local file system driver.
- The local file system driver passes the data back to the server 26.
- The server 26 passes the data to the lower-level network drivers for transmission back to the client computer 28.

Socket-based Applications

Socket-based applications are supported through the Windows Socket Provider (WinSock). Figure 3 shows the relationship between various modules and TCP/IP.

A Socket based user application that would like to provide a service on port P (pre-advertised) to all clients would open up a socket through WinSock and listen on the socket for connection requests. This would translate as an address object with TCB structures listening on that port with the server's IP address(es) and a wild-card IP address to denote client addresses.

When a client requests a connection, a frame comes in on one of the NICs 28 and 30 (with the client's IP address and connecting port #) and is tied to one of these TCB structures. TCP calls across TDI 18 to indicate to the socket provider, which in turn calls into the User's service application for acceptance. Once accepted, frames are received and sent on the NIC involved in the connection. Though the physical NIC through which the connection data is routed can change over time, the IP addresses and port numbers don't change and lend themselves as connection context for event tracing.

The user application requests a Send to the socket service provider with a pointer to the data. The socket provider, namely AFD 32 will lock the pages in memory and request TCP 14 across TDI 18 to send. TCP 14 cuts the requests to frames and calls the miniport driver 34 corresponding to the NIC through which the data needs to be transmitted. The Miniport driver sends the frames and calls to TCP 14 to complete each frame-send event. The TCP requests are processed asynchronously as a rule, and could happen in the context of the system thread dispatched by the socket provider or a DPC (Deferred Procedure Call) from Ndis.

In the case of receive, the miniport driver 34 services the interrupt and through Ndis 20, queues a DPC to process the frame. This DPC identifies the protocol stack and calls the appropriate Receive Event handler. When this thread executes TCP Receive routine in its context, the data is indicated up or is filled in pre-posted receive buffers from the application.

Event Tracing Sends / Receives

A TCP Send needs to be traced at the end of the send. The end of a send is marked by the processing of an ACK(acknowledgement) from the other connection endpoint corresponding to the last byte of the send. Through TDI, TCP receives an IoRequestPacket (IRP) from AFD with a pointer to a locked user buffer / locked set of pages from a file's cached view. TCP creates a Send-Request structure, which caches this IRP, splits the data into frames, and sends them across. When the last ack (ack acknowledging the last byte sent) arrives, part of receive processing involves queuing the corresponding Send-Request for completion. When the completion queue is processed, the cached IRP is completed to the upper layer.

Between the initiation of the Send and the completion, several copies of the data could get transmitted due to retransmissions, or the send could get cancelled, in which case, a completion doesn't occur. In Fig 4., the timeline of events during a send is explained. Since only the completion is traced, the use of the NIC to send out the said number of bytes is guaranteed. Also, as far as the user application is concerned, the Send through Tdi completed only when the IRP is completed in the socket driver (which may

wake up the blocked thread in the case of synchronous i/o or trigger the appropriate event in the case of async i/o).

Tracing receives is more complex than Sends, in the sense that tracing information needs to be generated at more points than one. In the case of receive, we should not care if the TCP protocol sends out acks or if this is only part of a receive which got cancelled from the other end point. The number of bytes received must be accounted for exactly. We will first take the case of a pre-posted receive and how it is traced for capacity planning purposes.

In the case of pre-posted receives, TCP receives an IRP through TDI to receive a certain size. TCP caches this IRP in a Receive-Request structure. When a chunk of a certain size of data is received (could be less than the requested size, to improve latency) TCP completes the request with the then-available number of bytes and the appropriate buffers. If more needs to be received, the receiver (say application through socket interface) posts more receive-IRPs which are completed as frames are received. In Fig 5., the receive completion and trace timing is explained.

It is possible to receive when no receives are posted. In such cases, the data is indicated to the receiver as soon as the first frame is received. If any more data needs to be received, TCP receives a piggybacked Irp. TCP generates a CP trace in this indicate path to accurately account for the accepted number of bytes.

Data Collection Issues

WINDOWS NT brand operating system uses a packet oriented I/O model for performing I/O operations to disks as well as to network devices. Whenever a user application or service posts a Send/Receive request, an IRP is created and sent to TCP. Typically the IRP is filled with the context of the thread requesting the operation. In the case of sends, it is possible to identify the correct user thread to charge the bandwidth utilization. In the case of receives, a DPC is generated in NDIS, which doesn't run in the context of any user thread. With respect to receives through the indicate-path described above, when the data is accepted without requesting more receives through IRPs, it is not

possible to make a correspondence. In such cases however, the port information that is provided is useful in identifying the service.

What is Instrumented?

1. Send Complete event when a TCP send request is complete (when an ACK for the last byte of the Send Request is received). The source address, destination address, source port number, destination port number, bytes transmitted are also recorded. Events are automatically time-stamped by the trace logger.
2. Receive Indicate event when incoming data is indicated to the upper layers. The source address, destination address, source port and destination port numbers, the size of data received and the process Id of the Process that is being indicated by TCP.
3. Receive Complete event (when a receive-Irp is completed with data). Similar information is collected.

These three trace points cover the majority of the meaningful TCP traffic in the system. It is important to keep in mind that some TCP traffic is not accounted for by this instrumentation. For example, retransmissions from packet loss, receiving IP control msgs (like ICMP etc.).

Instrumentation Overhead

In comparison to other kernel events such as thread create or delete, network events are very high frequency events. As a result, extraordinary care has been taken to minimize the overhead of trace instrumentation. The data being collected is primarily from the Transport Control Block (TCB) structure. The fields in the TCB structure are arranged to make the data relevant to capacity planning in one contiguous block. This allows direct copying from the TCB structure to the Trace Buffers without having to make any intermediary copies. According to measurements obtained during preliminary testing, a network event uses about 128 x86 instructions and logs 24 bytes of data. The actual results may vary, however.

Analysis of sample Traces

Appendix A provides a sample kernel trace fragment from a TCP Send test, translated into readable text format. Each row in the table shows an event instance. Each event instance is described by the fixed header providing the event name, Thread ID that is causing the event, system clock time when the event happened, the kernel and user mode CPU time for the thread. Additional columns in the table show the event specific data associated with each event.

The tests were started after starting up the Trace logger using a command line utility called tracelog.exe. The trace shows the process start and end of the tracelog.exe. Next there is a process start for the TCP send test program (nttcp.exe). Immediately following that, the Tcp Send events can be seen. The source IP address/port and destination IP address/port can also be seen.

The size of transfers is 8K bytes. The thread that's actually performing the send is (thread Id 0, a system thread). However, the process Id that was saved when the connection was created is 1C0, recorded with every send event. This process Id 1C0 corresponds to the nttcp.exe program that initiated the sends. Hence, it can be seen that the network traffic can be charged to processes properly from the traces.

While the TCPSend events triggered on the nttcp connection were in progress, an HTTP request for a webpage happened (shown in italics) and through threadID 39C, this request was handled by the IIS running on port 80 and 3 files were sent out to the HTTP remote browser. The HTTP transaction happened through a keep-alive connection. These events were charged to process 382 (InetInfo.exe).

Possible uses of network traces

Classification by PID:

In summary, since stack event trace generated incorporates PID, it is possible to charge network traffic to a specific process or kernel mode service. Other possible modes of classification, such as per-service, per-NIC, per-remote-request or per-client are indicated in paragraphs below.

Classification of network utilization per Service:

From post-processing the collected trace information, it is possible to classify the bandwidth utilization by application / service. Services and user applications / connections can be characterized using the 4-tuple (SAddr, DAddr, Sport, Dport). The traces collected for the specific port show the utilization for a particular service. From the traces shown, it can be observed that (HTTP) web service active on port 80 can be charged for the receives and sends in italics.

Classification of network utilization per NIC:

Using tools such as IPConfig, it is possible to identify NICs and assigned IP addresses. Parsing the collected trace for a specific IP Source Address gives all the traffic for that particular NIC. It is possible that data rerouting can happen when a transfer is in progress. In such a case, TCP receives an indication and a special stack event trace is generated.

Classification of System resource utilization per remote request / client:

Since Disk I/O events are generated in the context of the process, and remote requests are charged to the same process (through stack event traces), it is possible to identify the running service to charge disk i/o operations to. Based on Destination IP addr, this can be further classified per client of the service.

Conclusion

With the introduction of Event Tracing for WINDOWS NT 5.0 brand operating system, the resource consumption of CPU, Memory (Page faults), Disk I/O and Network and be charged to applications or Services. This will make the task of capacity planning client/server applications running on WINDOWS NT brand operating system servers easier and more accurate.

We claim:

1. A method of tracing data traffic on a network, the method comprising: at the transport layer of a protocol stack residing on a first device in the network, detecting a transmission or receipt of data to or from a second device on the network; and in response to the transmission or receipt being detected, recording the transmission or receipt as an entry in a trace log, wherein the trace log is accessible to determine the volume of data traveling over a network.

2. The method of claim 1, wherein the protocol stack is a TCP/IP stack.

3. The method of claim 1, wherein the detection step further comprises the step of detecting the presence of an input/output packet representing the transmission or receipt.

4. A method of tracing a transmission of data over a computer network comprising: detecting the presence of an input/output packet requesting a transmission; searching the input/output request packet to determine the identity of the process that created the input/output request packet; and storing in a trace log an entry representing the transmission, wherein the entry comprises the identity of the process, and wherein the trace log is accessible to determine the volume of data being transmitted over the network.

5. The method of claim 4, further comprising: detecting an acknowledgment of the transmission; and in response to the detection of the acknowledgment, storing in the trace log an entry representing the completion of the transmission.

6. A method of tracing a receipt of data from a computer network comprising: detecting the presence of a packet for an input/output connection to a port; searching the packet to determine the identity of the process that created the packet; and in response to the detection of a receipt of data at the port, storing in a trace log an entry

representing the receipt of the data, wherein the entry comprises the process identification, and wherein the trace log is accessible to determine the volume of the data being transmitted over the network.

7. The method of claim 6, further comprising: creating a connection object representing the opening of the port connection by the process; copying the process identification from the connection object into a transport control block associated with the port; and in response to the detection of the receipt of data at the port, copying the process identification into the trace log.

8. The method of claim 7, further comprising: copying the process identification from the connection object into the transport control block so that the process identification is contiguous with the rest of the data in the transport control block.

9. The method of claim 8, further comprising: detecting the presence of an input/output request packet indicating that the data receipt is complete; and in response to the detection of the completion input/ouput request packet, making an entry representing the receipt of the data into a trace log.

10. A facility for tracing data traffic on a network, the facility comprising: an identifying means for identifying a process causing a transmission or receipt of a communication via the network; and a logging means in communication with the identifying means for logging and event, wherein the event comprises the identification the process and wherein the logging means is useable to determine the volume of data traveling over the network.

11. The apparatus of claim 10 wherein the identifying means further comprises means for communicating with a transport layer of a protocol stack.

12. A computer-readable medium having stored thereon computer-executable instructions for performing steps comprising: at the transport layer of a protocol stack residing on a first device in the network, detecting a transmission or receipt of data to or from a second device on the network; and in response to the transmission or receipt being

detected, recording the transmission or receipt as an entry in a trace log, wherein the trace log is accessible to determine the volume of data traveling over a network.

13. The computer-readable medium of claim 12, wherein the protocol stack is a TCP/IP stack.

14. The computer-readable medium of claim 12, having further computer-executable instructions for performing the step of detecting the presence of an input/output packet representing the transmission or receipt.

15. A computer-readable medium having stored thereon computer-executable instructions for performing steps comprising: detecting the presence of an input/output packet requesting a transmission; searching the input/output request packet to determine the identity of the process that created the input/output request packet; and storing in a trace log an entry representing the transmission, wherein the entry comprises the identity of the process, and wherein the trace log is accessible to determine the volume of data being transmitted over the network.

16. The computer-readable medium of claim 15, having further computer-executable instructions for performing the step of detecting the presence of the input/output packet at the transport layer of a protocol stack.

17. The computer-readable medium of claim 15, having further computer-executable instructions for performing the step of detecting an acknowledgment of the transmission; and in response to the detection of the acknowledgment, storing in the trace log an entry representing the completion of the transmission.

18. A computer-readable medium having stored thereon computer-executable instructions for performing the steps comprising: detecting the presence of a packet for an input-output connection to a port; searching the packet to determine the identity of the process that created the packet; and in response to the detection of a receipt of data at the port, storing in a trace log an entry representing the receipt of the data, wherein the entry

comprises the process identification, and wherein the trace log is accessible to determine the volume of the data being transmitted over the network.

19. The computer-readable medium of claim 18, having further computer-executable instructions for performing the steps of: creating a connection object representing the opening of the port connection by the process; copying the process identification from the connection object into a transport control block associated with the port; and in response to the detection of the receipt of data at the port, copying the process identification into the trace log.

20. The computer-readable medium of claim 18, having further computer-executable instructions for performing the step of copying the process identification from the connection object into the transport control block so that the process identification is contiguous with the rest of the data in the transport control block.

21. The computer-readable medium of claim 18, having further computer-executable instructions for performing the steps of: detecting the presence of an input/output request packet indicating that the data receipt is complete; and in response to the detection of the completion input/output request packet, storing in the trace log an entry representing the receipt of the data.

Appendix A:

Event Name	TID	Clock (ms)	Kt	Ut	Parent TID	Parent PID	Sz	Image Name		
ProcessStart	1C0	1124570445	2	0	10C	34C		Tracelog.exe		
ThreadStart	1C0	1124570445	2	0	1C0	10C				
ThreadEnd	1C0	1124570492	3	0	1C0	10C				
ProcessEnd	1C0	1124570492	3	0	10C	34C		Tracelog.exe		
DiskWriteIo	14	1124571070	0	0	2	67	300			
ProcessStart	2AC	1124572773	36	28	1C0	288		CMD.exe		
ThreadStart	2AC	1124572773	36	28	10C	1C0				
ThreadStart	10C	1124572898	0	2	364	1C0				
Event Name	**TID**	**Clock (ms)**	**Kt**	**Ut**	**Source Addr**	**Dest Addr**	**Sport**	DPort	**Size**	**PID**
TcpSend	0	1124572976	26425	0	172.31.249.34	172.31.255.147	5010	5020	8192	1C0
TcpSend	0	1124572976	26425	0	172.31.249.34	172.31.255.147	5010	5020	8192	1C0
TcpSend	0	1124572976	26425	0	172.31.249.34	172.31.255.147	5010	5020	8192	1C0
TcpSend	0	1124572992	26426	0	172.31.249.34	172.31.255.147	5010	5020	8192	1C0
TcpSend	0	1124572992	26426	0	172.31.249.34	172.31.255.147	5010	5020	8192	1C0
TcpSend	0	1124573008	26426	0	172.31.249.34	172.31.255.147	5010	5020	8192	1C0
TcpSend	0	1124573008	26426	0	172.31.249.34	172.31.255.147	5010	5020	8192	1C0
TcpSend	0	1124573023	26427	0	172.31.249.34	172.31.255.147	5010	5020	8192	1C0
TcpRecv	***0***	***1124573093***	***78293***	***1***	***172.31.249.34***	***172.31.254.11***	***80***	***6437***	***531***	***382***
TcpRecv	***0***	***1124573164***	***78294***	***3***	***172.31.249.34***	***172.31.254.11***	***80***	***6437***	***164***	***382***
TcpSend	***39C***	***1124573198***	***78296***	***0***	***172.31.249.34***	***172.31.254.11***	***80***	***6437***	***1507***	***382***
TcpSend	***39C***	***1124573231***	***78303***	***0***	***172.31.249.34***	***172.31.254.11***	***80***	***6437***	***9236***	***382***
TcpSend	***39C***	***1124573362***	***78304***	***0***	***172.31.249.34***	***172.31.254.11***	***80***	***6437***	***6728***	***382***
TcpSend	0	1124573570	26460	0	172.31.249.34	172.31.255.147	5010	5020	8192	1C0
TcpSend	0	1124573586	26461	0	172.31.249.34	172.31.255.147	5010	5020	8192	1C0
TcpSend	0	1124573586	26461	0	172.31.249.34	172.31.255.147	5010	5020	8192	1C0
TcpSend	0	1124573601	26462	0	172.31.249.34	172.31.255.147	5010	5020	8192	1C0
TcpSend	0	1124573601	26462	0	172.31.249.34	172.31.255.147	5010	5020	8192	1C0
TcpSend	0	1124573617	26463	0	172.31.249.34	172.31.255.147	5010	5020	8192	1C0
TcpSend	0	1124573617	26463	0	172.31.249.34	172.31.255.147	5010	5020	8192	1C0
TcpSend	0	1124573633	26464	0	172.31.249.34	172.31.255.147	5010	5020	8192	1C0
TcpSend	0	1124573648	26465	0	172.31.249.34	172.31.255.147	5010	5020	8192	1C0
TcpSend	0	1124573648	26465	0	172.31.249.34	172.31.255.147	5010	5020	8192	1C0

Event Name	TID	Clock (ms)	Kt	Ut	Parent TID	Parent PID	Sz	Image Name
ThreadEnd	364	1124573758	0	0	364	1C0		
ThreadEnd	10C	1124573789	1	3	10C	1C0		
ProcessEnd	10C	1124573789	1	3	1C0	288		nttcp.exe

Legend: Kt - Kernel Mode Time

Ut - User Mode CPU Time

PID - Process ID

TID - Thread ID

The following are the extended record fields for the respective events:

- Process start - new Process Id, its Parent's Process id
- Process end - current Process Id, its Parent's Process id, the image filename that it was running
- Thread start - new thread Id, its Process Id
- Thread end - current thread Id being terminated, its Process Id.
- I/O read, I/O write - The signature of the disk where the I/O operation was done, the transfer size.
- TCPSend - Source Address, Destination Address, Source port, Destination port, Size, ProcessId

ABSTRACT

A method of tracing data on a network that detects receive and send events in a protocol stack is provided. The method provides key information for capacity planners for correctly charging network traffic to the individual services and applications. The TCP/IP stack is instrumented at the transport layer, so that Input/Output Request packets (IRP) representing sends and receives can be detected as they pass through the stack. When such packets are detected an appropriate send or receive event is recorded in a trace log.

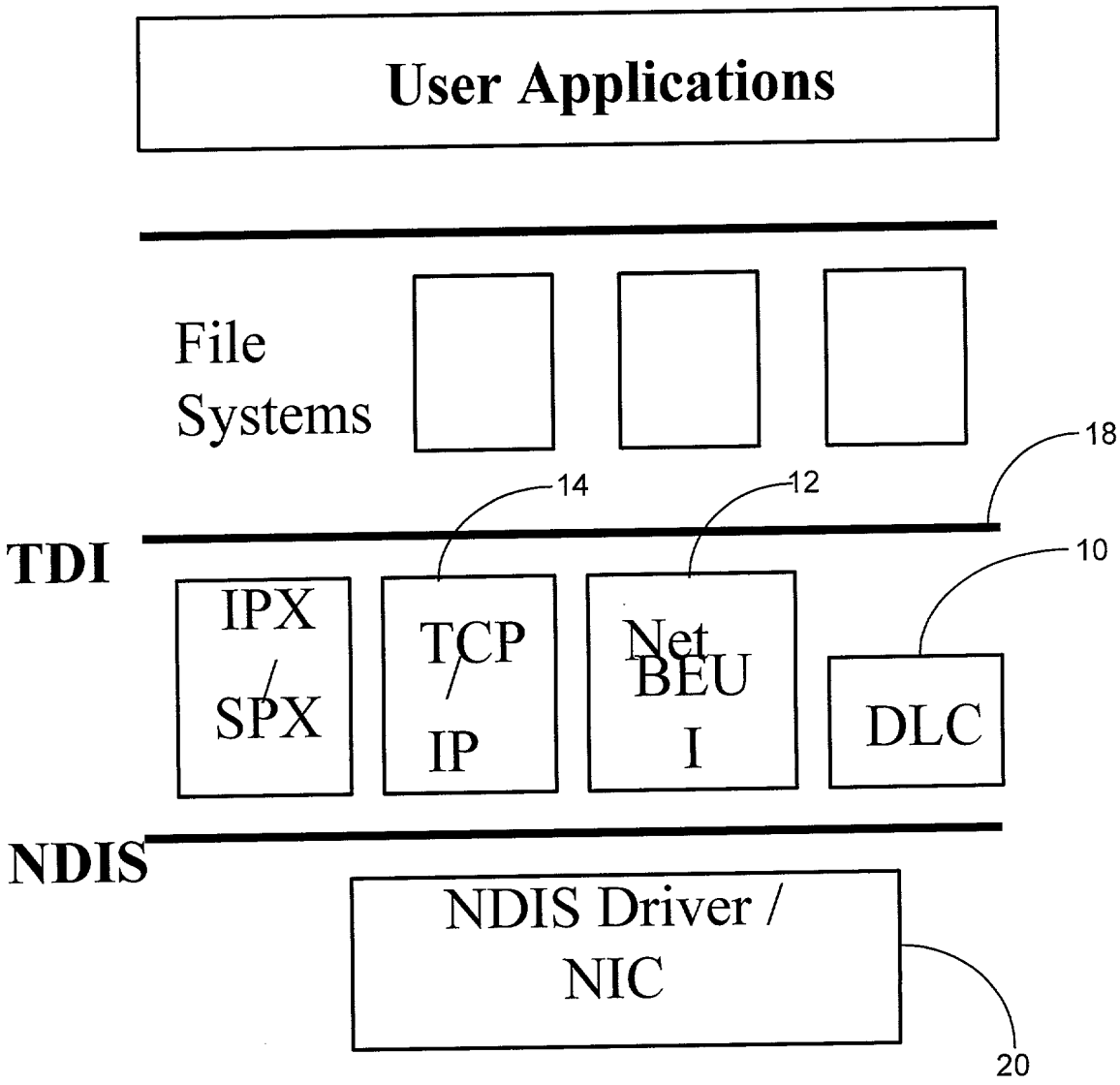

User Applications
File Systems
TDI
IPX / SPX
TCP / IP
Net BEU I
DLC
NDIS
NDIS Driver / NIC
14
12
18
10
20

FIG. 1

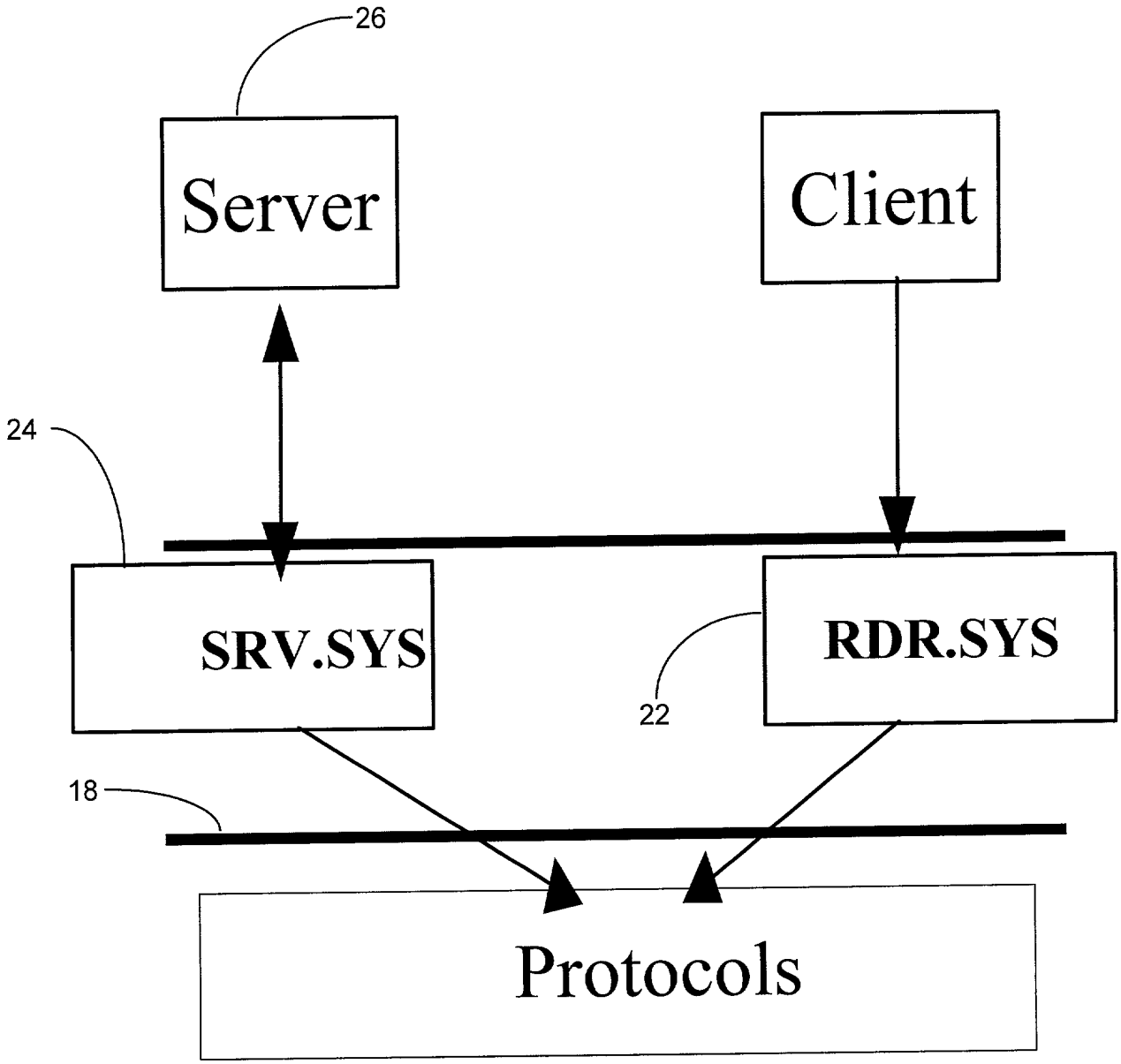


FIG. 2

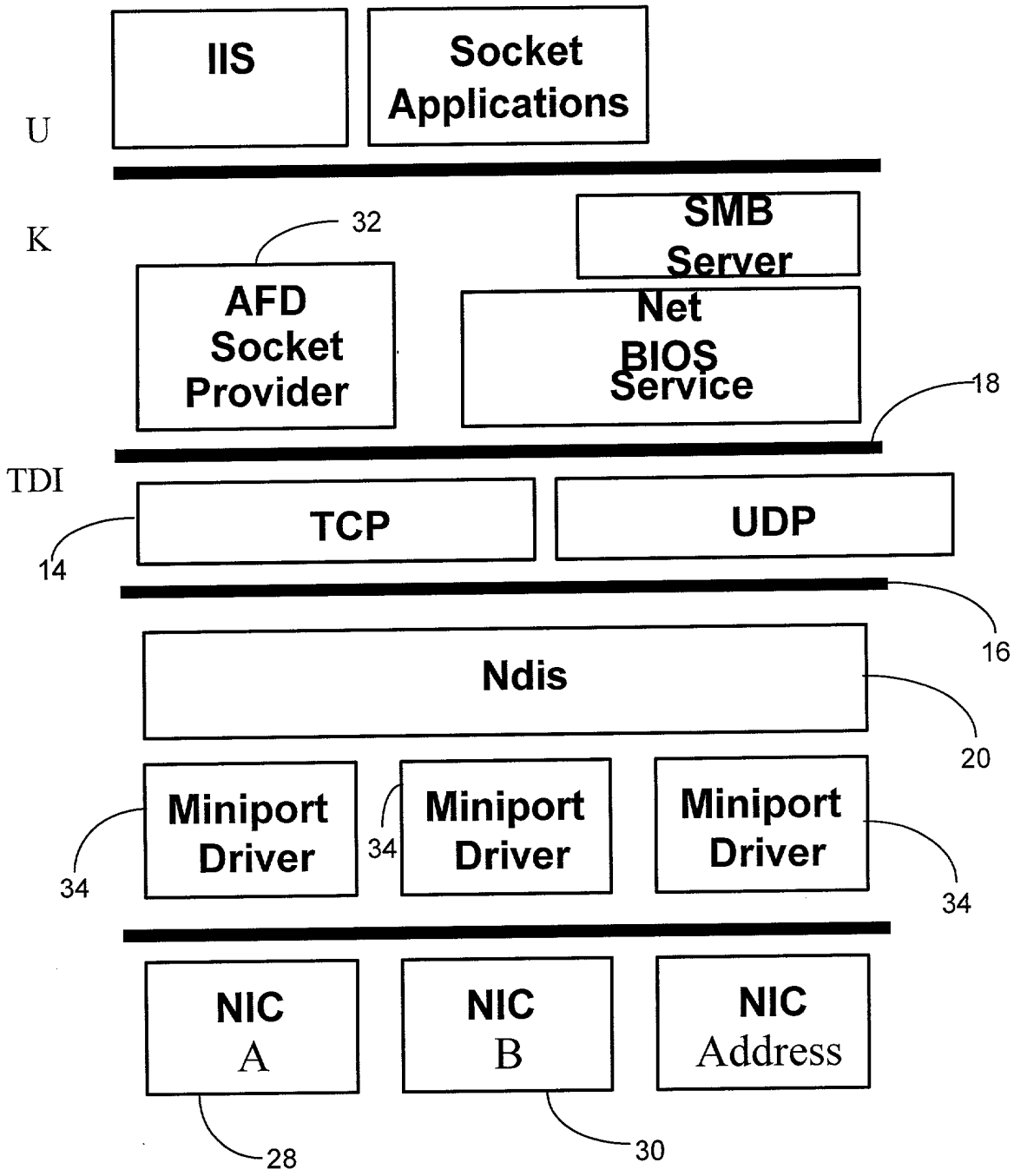

IIS
Socket Applications
U
K
32
SMB Server
AFD Socket Provider
Net BIOS Service
18
TDI
TCP
UDP
14
16
Ndis
20
Miniport Driver
34
Miniport Driver
34
Miniport Driver
34
NIC A
NIC B
NIC Address
28
30

FIG. 3

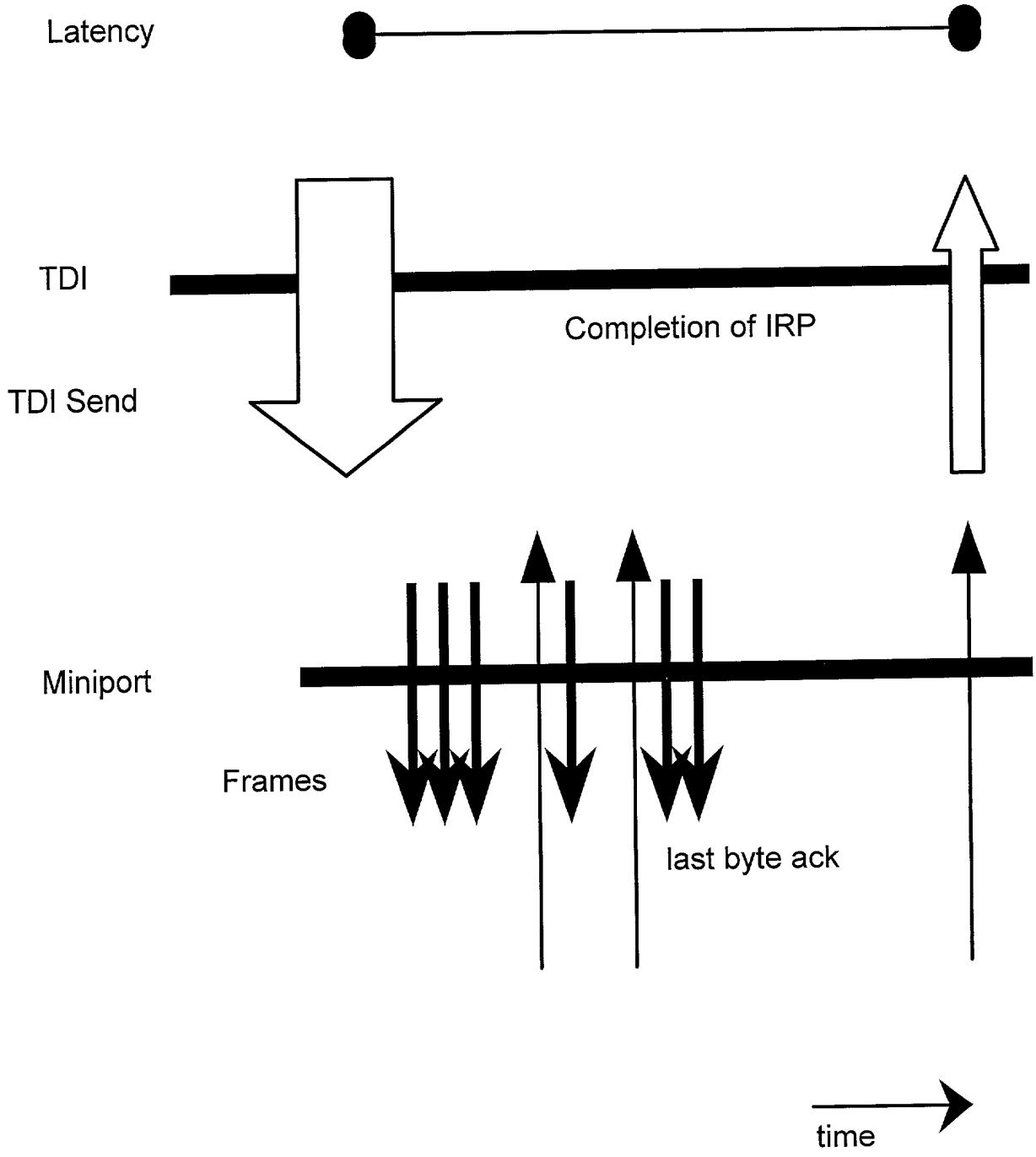

Latency
TDI
Completion of IRP
TDI Send
Miniport
Frames
last byte ack
time

FIG. 4

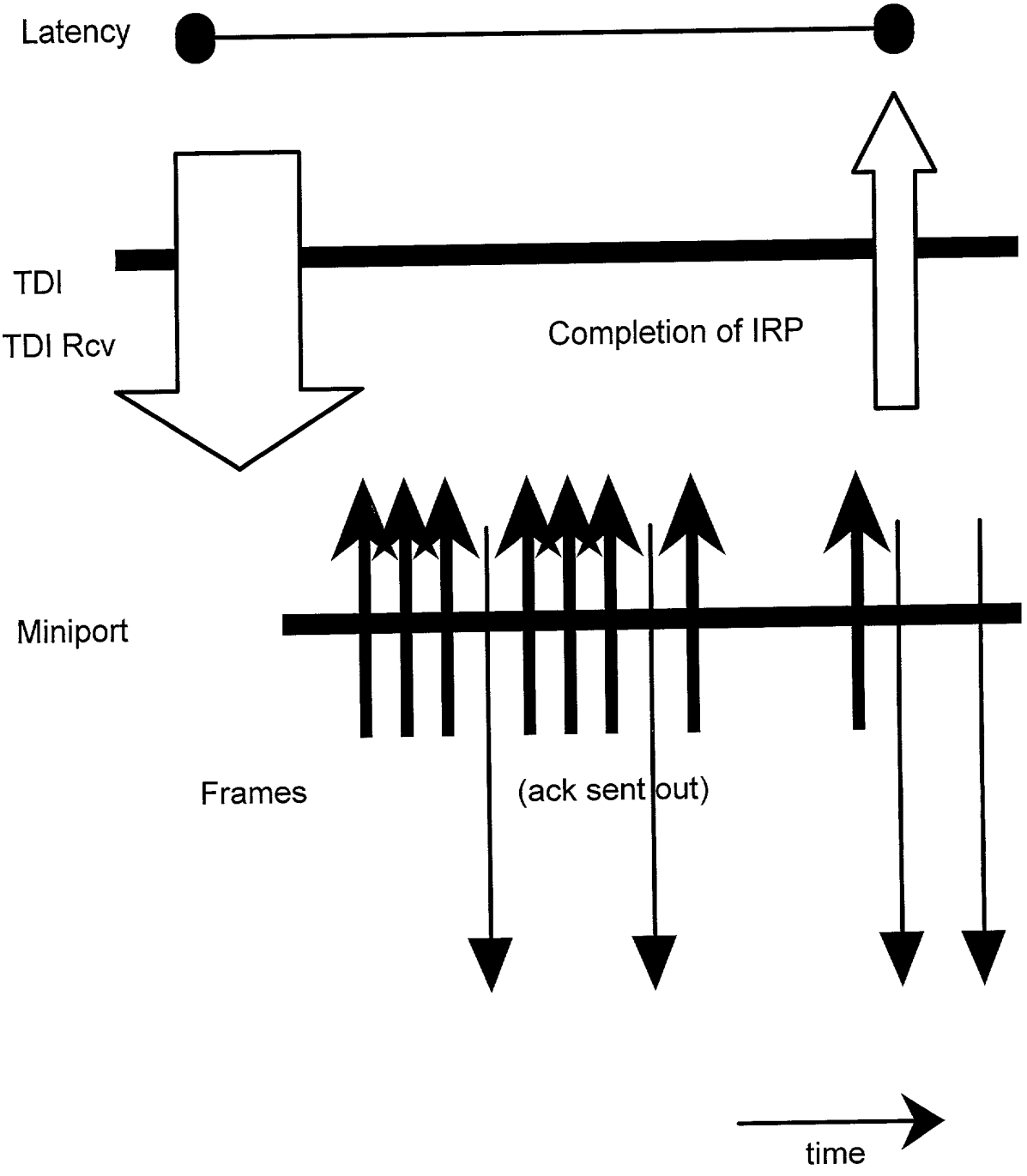

Latency
TDI
TDI Rcv
Completion of IRP
Miniport
Frames
(ack sent out)
time

FIG. 5

PATENT
Attorney's Docket No. 202269

COMBINED DECLARATION AND POWER OF ATTORNEY

As below named inventor, I hereby declare that

This declaration is of the following type:
☒ original ☐ design ☐ supplemental
☐ national stage of PCT
☐ divisional ☐ continuation ☐ continuation-in-part

My residence, post office address, and citizenship are as stated below next to my name. I believe I am the original, first, and sole inventor *(if only one name is listed below)* or an original, first, and joint inventor *(if plural names are listed below)* of the subject matter which is claimed and for which a patent is sought on the invention entitled:

METHOD OF TRACING DATA TRAFFIC ON A NETWORK

the specification of which:
☒ is attached hereto.
☐ was filed on as Serial No. and was amended on_____. *(if applicable).*
☐ was filed by Express Mail No. *as Serial No. not known yet,* and was amended on _____ *(if applicable).*
☒ was described and claimed in PCT International Application No. _____ filed on _____ and as amended under PCT Article 19 on _____ *(if any).*

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claim(s), as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, Code of Federal Regulations, § 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, § 119 of any foreign application(s) for patent or inventor's certificate or of any PCT international application(s) designating at least one country other than the United States of America listed below and have also identified below any foreign application(s) for patent or inventor's certificate or any PCT international application(s) designating at least one country other than the United States of America filed by me on the same subject matter having a filing date before that of the application(s) of which priority is claimed.

COUNTRY	APPLICATION	DATE OF FILING (day,**month**,year)	PRIORITY CLAIMED UNDER 35 USC 119			
				YES		NO
				YES		NO
				YES		NO

I hereby claim the benefit pursuant to Title 35, United States Code, § 119(e) of the following United States provisional application(s):

PRIOR U.S. PROVISIONAL APPLICATIONS CLAIMING THE BENEFIT UNDER 35 USC 119(e)	
APPLICATION NO.	DATE OF FILING
60/141,581	June 23, 1999

I hereby claim the benefit under Title 35, United States Code, § 120 of any United States application(s) or PCT international application(s) designating the United States of America that is/are listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in that/those prior application(s) in the manner provided by the first paragraph of Title 35, United States Code, § 112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, § 1.56 which occurred between the filing date of the prior application(s) and the national or PCT international filing date of this application.

PRIOR U.S. APPLICATIONS OR PCT INTERNATIONAL APPLICATIONS DESIGNATING THE U.S. FOR BENEFIT UNDER 35 USC 120					
U.S. APPLICATIONS			Status *(check one)*		
U.S. APPLICATIONS	U.S. FILING DATE		PATENTED	PENDING	ABANDONED
1. 0 /					
2. 0 /					
3. 0 /					
PCT APPLICATIONS DESIGNATING THE U.S.			Status *(check one)*		
PCT APPLICATION NO.	PCT FILING DATE	U.S. SERIAL NOS. ASSIGNED (if any)	PATENTED	PENDING	ABANDONED
4.					
5.					
6.					

DETAILS OF FOREIGN APPLICATIONS FROM WHICH PRIORITY CLAIMED UNDER 35 USC 119 FOR ABOVE LISTED U.S./PCT APPLICATIONS				
ABOVE APPLN. NO.	COUNTRY	APPLICATION NO.	DATE OF FILING (day,**month**,yr)	DATE OF ISSUE (day,**month**,yr)
1.				
2.				
3.				
4.				

As a named inventor, I hereby appoint the following attorneys to prosecute this application and transact all business in the Patent and Trademark Office connected therewith.

Berton Scott Sheppard, Reg. 20922
James B. Muskal, Reg. 22797
Dennis R. Schlemmer, Reg. 24703
Gordon R. Coons, Reg. 20821
John E. Rosenquist, Reg. 26356
John W. Kozak, Reg. 25117
Charles S. Oslakovic, Reg. 27583
Mark E. Phelps, Reg. 28461
H. Michael Hartmann, Reg. 28423
Bruce M. Gagala, Reg. 28844
Charles H. Mottier, Reg. 30874
John Kilyk, Jr., Reg. 30763
Robert F. Green, Reg. 27555
John B. Conklin, Reg. 30369
James D. Zalewa, Reg. 27848
John M. Belz, Reg. 30359
Brett A. Hesterberg, Reg. 31837
Jeffrey A. Wyand, Reg. 29458

Paul J. Korniczky, Reg. 32849
Pamela J. Ruschau, Reg. 34242
Steven P. Petersen, 32927
John M. Augustyn, Reg. 33589
Christopher T. Griffith, Reg. 33392
Wesley O. Mueller, Reg. 33976
Jeremy M. Jay, Reg. 33587
Jeffrey B. Burgan, Reg. 35463
Eley O. Thompson, Reg. 36035
Mark Joy, Reg. 35562
Allen E. Hoover, Reg. 37354
David M. Airan, Reg. 38811
Michael H. Tobias, Reg. 32948
Xavier Pillai, Reg. 39799
Y. Kurt Chang, Reg. 41397
Gregory C. Bays, Reg. 40505
Carol Larcher, Reg. 35243

Steven H. Sklar, Reg. 42154
M. Daniel Hefner, Reg. 41826
Thomas A. Belush, Reg. 37090
Kenneth P. Spina, Reg. 43927
Gary R. Jarosik, Reg. 35906
Song Zhu, Reg. 44420
Jeffery J. Makeever, Reg. 37390
Salim A. Hasan, Reg. 38175
Richard A. Wulff, Reg. 42238
Jamison E. Lynch, Reg. 41168
Rattan Nath, Reg. 43827
Robert M. Gould, Reg. 43642
Kevin L. Wingate, Reg. 38662
David J. Schodin, Reg. 41294
Paul L. Ahern, Reg. 17020
Theodore W. Anderson, Reg. 17035
Noel I. Smith, Reg. 18698
Katie E. Sako, Reg. 32628
Daniel D. Crouse, Reg. 32022

I further direct that correspondence concerning this application be directed to LEYDIG, VOIT & MAYER, LTD., Two Prudential Plaza, Suite 4900, 180 North Stetson, Chicago, Illinois 60601-6780, Telephone (312) 616-5600.

I hereby declare that all statements made herein of my own knowledge are true, that all statements made on information and belief are believed to be true, that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full name of **sole or first inventor:** Melur K. Raghuraman

Inventor's signature _____

Date 1/18/2000

Country of Citizenship: India

Residence: 2527 238th Court NE, Redmond, Washington 98053

Post Office Address: Same as above

Full name of **second joint inventor,** if any: Venkataraman Ramanathan

Inventor's signature _____

Date 1/20/2000

Country of Citizenship: India

Residence: 1735 233rd Place NE, Redmond, Washington 98053

Post Office Address: Same as above

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:

RAGHURAMAN et al.

For: METHOD OF TRACING DATA TRAFFIC ON A NETWORK

DECLARATION OF MELUR RAGHURAMAN UNDER 37 C.F.R. § 1.56

Honorable Commissioner of
Patent and Trademarks
Washington, D.C. 20231

Dear Sir:

I, Melur Raghuraman, do hereby declare that:

1. I am an inventor of the invention described and claimed in the accompanying patent application entitled "METHOD OF TRACING DATA TRAFFIC ON A NETWORK" that is being filed concurrently herewith (Attorney Docket No. 202269).

2. The provisional application to which the present application claims priority (U.S. provisional application 60/140,581) contains a paper entitled "NETWORK PERFORMANCE MONITORING IN WINDOWS NT." This paper was presented at the 1998 International Conference on the Computer Measurement Group (CMG) held December 7-11, 1998 in Anaheim, California.

3. At the bottom of the sixth page of the paper identified in paragraph 2 above is a copyright notice dated 1997. THE DATE INCLUDED IN THIS NOTICE IS INCORRECT. The paper was first released to the public at the conference identified in paragraph 2 above. The paper was not made public prior to December 7, 1998.

4. The language in the notice identified in paragraph 3 above was erroneously copied from an unrelated document and the mistake in the copyright notice date went undetected until after the provisional application identified in paragraph 2 above was filed.

I hereby declare that all statements made herein of my own knowledge are true, that these statements were made with the knowledge that willful false statements and the like so made are publishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

1/18/2000
Date

Melur Raghuraman

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:

RAGHURAMAN et al.

For: METHOD OF TRACING DATA TRAFFIC ON A NETWORK

DECLARATION OF VENKAT RAMANATHAN UNDER 37 C.F.R. § 1.56

Honorable Commissioner of
Patent and Trademarks
Washington, D.C. 20231

Dear Sir:

I, Venkat Ramanathan, do hereby declare that:

1. I am an inventor of the invention described and claimed in the accompanying patent application entitled "METHOD OF TRACING DATA TRAFFIC ON A NETWORK" that is being filed concurrently herewith (Attorney Docket No. 202269).

2. The provisional application to which the present application claims priority (U.S. provisional application 60/140,581) contains a paper entitled "NETWORK PERFORMANCE MONITORING IN WINDOWS NT." This paper was presented at the 1998 International Conference on the Computer Measurement Group (CMG) held December 7-11, 1998 in Anaheim, California.

3. At the bottom of the sixth page of the paper identified in paragraph 2 above is a copyright notice dated 1997. THE DATE INCLUDED IN THIS NOTICE IS INCORRECT. The paper was first released to the public at the conference identified in paragraph 2 above. The paper was not made public prior to December 7, 1998.

4. The language in the notice identified in paragraph 3 above was erroneously copied from an unrelated document and the mistake in the copyright notice date went undetected until after the provisional application identified in paragraph 2 above was filed.

I hereby declare that all statements made herein of my own knowledge are true, that these statements were made with the knowledge that willful false statements and the like so made are publishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

1/20/2000
Date

Venkat Ramanathan